Gunrock 2.0: A User Adaptive Social Conversational System

Kaihui Liang[*], **Austin Chau, Yu Li, Xueyuan Lu, Dian Yu**[†],
Mingyang Zhou, Ishan Jain, Sam Davidson[◇], **Josh Arnold, Minh Nguyen, Zhou Yu**[‡]
Department of Computer Science
[◇]Department of Linguistics
University of California, Davis
Davis, CA 95616

Abstract

Gunrock 2.0 is built on top of Gunrock with an emphasis on user adaptation. Gunrock 2.0 combines various neural natural language understanding modules, including named entity detection, linking, and dialog act prediction, to improve user understanding. Its dialog management is a hierarchical model that handles various topics, such as movies, music, and sports. The system-level dialog manager can handle question detection, acknowledgment, error handling, and additional functions, making downstream modules much easier to design and implement. The dialog manager also adapts its topic selection to accommodate different users' profile information, such as inferred gender and personality. The generation model is a mix of templates and neural generation models. Gunrock 2.0 is able to achieve an average rating of 3.73 at its latest build from May 29th to June 4th.

1 Introduction

Protecting user privacy is an essential task which many companies are taking seriously, this means many large, clean, real conversation data, such as customer service dialogs, are not publically available. While this protection should not be compromised, this still poses a significant challenge for the academic community in developing advanced dialog models. Current dialog corpora frequently used in dialog research, such as MultiWOZ(3) and PersuasionForGood (34) are all collected through role-play formats on crowdsourcing platforms. No matter how well the tasks are designed, these role-play datasets have limitations compared to those from real users who have real intent to interact with the dialog systems. Amazon has the foresight to help academia and industry solve this problem by launching the Amazon Alexa Prize for building engaging open-domain social conversational systems. The Amazon Alexa Prize provides a platform that attracts a large number of volunteer users with real intent to interact with the participating social conversational systems. Anyone in the United States who has an Alexa powered device, such as Amazon Echo, can interact with our system. This platform provides us an extensive and detailed dataset for developing a large-scale conversational system and conducting dialog system research.

We built Gunrock 2.0 based on our 2018 Alexa Prize-winning bot, Gunrock. Gunrock 2.0 extends the design idea of user adaptation. We made a number of contributions in open domain spoken language understanding, dialog management, and language generation.

[*]kaliang@ucdavis.edu
[†]dianyu@ucdavis.edu
[‡]joyu@ucdavis.edu

3rd Proceedings of Alexa Prize (Alexa Prize 2020).

We improved on several aspects of the previous Gunrock system: 1) Better understanding modules, including more fine-grained intent detection, and better-named entity detection and linking models; 2) A generation based acknowledgment model; 3) A user-adaptive dialog manager that controls topic selection; 4) A generation based follow-up request and response.

2 Related Work

Open-domain chatbots, such as the classic example Alice (33), aim to pass the Turing Test. In comparison, social chatbots require more in-depth communication skills with emotional support(27). Gunrock 2.0 extends Gunrock and utilizes state-of-the-art practices in both domains and emphasizes dynamic user adaptation conversations.

Many neural models (32) and reinforcement learning models (17) have been proposed for end-to-end dialog generation and understanding. With large datasets available, including Cornell Movie Dialogs (5) and Reddit [4], these models improve dialog performance using an end-to-end approach. Pre-training based dialog models such as ARDM (35) or DialogGPT (41) have been very popular as well. However, these methods still suffer from incoherent and generic responses (42).

In order to solve those problems, some research has combined rule-based and end-to-end approaches (21). Other relevant work leverages individual mini-skills and knowledge graphs (10). This combination of approaches enhances user experience and prolongs conversations. However, they are not flexible in adapting to new domains and cannot handle robustly opinion related requests. For example, in the 2017 Alexa Prize [5], Sounding Board (10) reported the highest weekly average feedback rating (one to five stars, indicating how likely the user is willing to talk to the system again) with a 3.37 across all conversations. For comparison, in 2018, our chatbot Gunrock reached 3.62 across all conversations.

Gunrock 2.0 takes full advantage of better understanding models, including better entity detection and linking, detailed intent detection, better acknowledgment, better user adaptation in dialog management and a generation model to handle out of domain follow-up questions. These novel concepts contributed to our rating of 3.73 from May 29th to June 4th using our latest build.

3 Architecture

We leveraged the Amazon Conversational Bot Toolkit (Cobot) (14) to build the system architecture. The toolkit provides a zero-effort scaling framework, allowing developers to focus on building a user-friendly bot. The event-driven system is implemented on top of the Alexa Skill Kit (ASK) with AWS Lambda function[6] as an endpoint. Each conversation turn is sent as an event request to the system. The Cobot framework uses a state manager interface that stores both user attributes and dialog state data to DynamoDB[7]. We also utilized Redis[8] and DynamoDB to build the internal system's knowledge base.

We will cover each system component in this section.

3.1 System Overview

Figure 1 depicts the social bot dialog system framework. Alexa Skill Kit provides the text of user utterances with timestep and confidence information through an automatic speech recognition (ASR) model. Our system takes the text input and generates text output in the Speech Synthesis Markup Language (SSML) format. Then ASK's built-in Text-To-Speech (TTS) service takes the text and generates speech output.

To avoid users talking about profane content, we detect profanity using Amazon's Offensive Speech Classifier toolkit (13) and our custom profanity list with regex. We inform the user of the inappropriateness of the topic and recommend an alternate subject to continue the conversation. Additionally,

[4] https://www.reddit.com/

[5] https://developer.amazon.com/alexaprize/2017-alexa-prize

[6] http://aws.amazon.com/lambda

[7] https://aws.amazon.com/dynamodb/

[8] https://redis.io

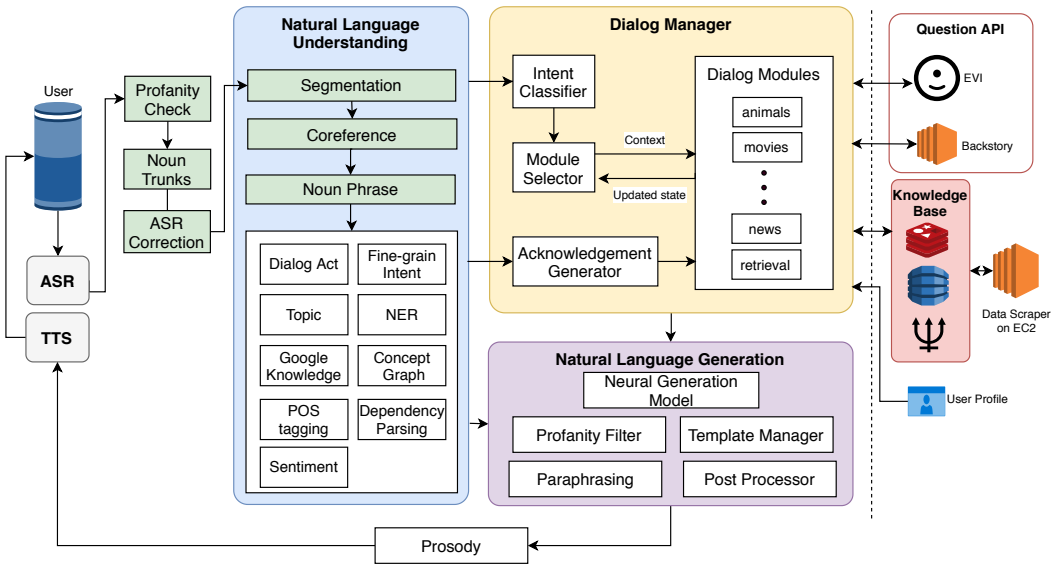

Figure 1: Gunrock System Architecture

if the ASR confidence score is low, we prompt the user to clarify their utterance. In both cases, we bypass the downstream NLU pipeline to reduce latency.

After ASR, the user input is processed by multiple NLU components. Figure 1 shows that there are 12 components in NLU. We use Cobot's built-in natural language processing pipeline, which supports multi-threaded task execution. There are four main steps in the NLU pipeline. First, the input utterance is segmented into multiple sentences. Then we solve coreference and detect noun phrases. Finally, user utterances are analyzed by several NLP components, as shown in Figure 1, in parallel. We will discuss the NLU components in detail in Section 3.3.

In the Dialog Manager, the Intent Classifier combines all NLU features and further classifies them into multiple intents. Module Selector then directs different user intents to corresponding dialog modules as described in 3.4.1. The dialog modules cover several specific topics, such as movies, sports, and animals. Each topic dialog module has its own dialog flow, and it adapts the flow to users' interests. We will explain this in more detail in Section 4. The dialog manager also contains an acknowledgment generator, which acknowledges the user's input depending on the context and user intents. We would describe it in Section 6.

We store our knowledge database offline on Redis and Dynamodb. We use Amazon EC2 to regularly scrape data from different sources to keep up with new content such as news and movie trivia.

Our NLG consists of manually curated templates, content retrieved from the internet, and responses generated by a neural generation model. For templates, we use a template manager to select templates based on the key assigned by topic modules. To ensure the appropriateness of content retrieved from the internet, we include a profanity checker to review the response's content. A post-processor is also included in NLG to modify the original topic dialog module response. Finally, we use Amazon SSML to format the prosody of the response. We will cover more details in 3.5.

3.2 Automatic Speech Recognition

Similar to Gunrock, we filter ASR outputs with a confidence threshold. The system asks the users to repeat their input instead of sending through the NLU and NLG pipeline if the ASR confidence is low. Even though improved ASR models achieve lower word error rates, recognizing the correct words is still challenging, mainly due to new phrases and lack of context. We apply ASR correction proposed by Gunrock by comparing noun trunk phonetic information to the noun phrases in the context domain, such as movie names, using Double Metaphone (20). In instances where we can unambiguously capture the error phrases using regular expression, we replace the erroneous phrases with the corrected phrases before the NLU pipeline processes them.

3.3 Nature Language Understanding

We follow the general NLU pipeline of Gunrock (36) to segment user input into semantic units after ASR correction and apply coreference resolution, constituency parsing, noun phrase extraction, named entity recognition, and sentiment analysis on each segment. We also add models such as sentence completion and dependency parsing for in-depth understanding and focus on improving each model with active learning strategies.

3.3.1 Gunrock NLU

Our NLU pipeline is similar to Gunrock. Specifically, we borrow the coreference resolution method with error fixes, constituency parsing for noun phrase extraction, and entity linking with Google Knowledge Graph [9] and Microsoft Concept Graph [10]. We improve other models detailed below.

3.3.2 Sentence Segmentation

Gunrock converts the sentence segmentation into a machine translation problem by training a sequence to sequence (s2s) (29) bidirectional LSTM model (12) using the Cornell Movie Quotes corpus (5). The source is the corpus after removing all the punctuation, and the target is to recover the sentence boundaries. With masking NER as pre-processing, maintaining original tokens as post-processing, and setting a timestop threshold, the model achieves 84.43% F1 score but performs worse on longer utterances. To improve the model performance, we experimented with different models, including transformer (31) as an s2s model and a sequence tagging model to predict sentence boundaries. With more annotated in-domain data and longer context, we achieve an F1 score of 91.9% on an annotated development set.

3.3.3 Dialog Act Prediction

Gunrock uses MIDAS (38) dialog act scheme for multi-label dialog act prediction in a hierarchical structure. For more robust dialog act prediction incorporating context, we use Hybrid-EL-CMP (40) by completing utterances and predicting dialog act with a combined distribution of the ellipsis and the automatically completed utterance. In addition, we improve our dialog act model by using active learning strategies (26). In specific, we extract utterances using regex to boost training examples for low-performance labels, and we also choose utterances with low prediction confidence. We retrain the model with an updated training corpus and keep improving model performances during deployment.

3.3.4 Fine-Grained Intents

We define extra intents as fine-grained intents that are used to complement our dialog acts. These fine-grained intents provide supplemental labeling of the user intents for our downstream tasks. We use regex to detect these intents based on the conversation we collected throughout the competition. For example, if users ask "`ask_back`" questions (e.g. "what about you? "), we would answer it according to the current topic. If users express their opinion, we use "`opinion_positive`" (e.g. "that's interesting") and "`opinion_negative`" (e.g. "it's boring") to understand user's opinion positivity. We also use "`ans_like`" (e.g. "I'm into movies") and "`ans_dislike`" (e.g. "I'm not a big fan of movies") to detect users' preferences towards a topic.

3.3.5 Noun Phrase Extraction

Semantic parsing such as AMR (15) and UCCA (1; 37) has been popular to facilitate language understanding tasks by extracting relationships among entities. However, they are expensive to annotate and hard to transfer to dialog domains for good performance. On the other hand, sequence tagging methods for NER detection have been well studied and are reliable with the benefit of large language models (7), but do not work well for regular nouns, which are essential for topic detection. Therefore, we follow the constituency parsing method of Gunrock to get noun phrases in different levels in the constituency tree.

[9] `https://developers.google.com/knowledge-graph/`

[10] `https://concept.research.microsoft.com/Home/Introduction`

However, even though such a method extracts noun phrases exhaustively and achieves high recall, it suffers from two problems: low precision; and lack of understanding of relationships. Therefore, we may not be able to detect the true topic with multiple noun phrases. To solve the first problem, we use a BiLSTM-CRF sequence tagging model to identify noun phrases that are important for understanding. We obtain training data for this model by filtering out non-informative noun phrases from the constituency parser. We referred to this model as the KeyPhrase model. Since the labels in the training data are noisy, we used entropy regularization (24) during training so that the model could reject inconsistent labels. For the second problem, we use the extended SCUD annotation scheme (6) and build a dependency parsing model on annotated dialog data to extract relationships among entities. The extracted noun phrases are used as input to entity linking for topic detection.

3.3.6 Entity Linking

Recognizing and linking entities such as movies, books, and people are critical in a dialog system. The problem of entity detection and linking is even more challenging in spoken settings than in written text because of the following problems. First, modeling context is harder because users can avoid questions and mention completely different things. However, modeling context is important to resolve whether Harry Potter is a "fictional character" or a "movie". Second, the input text is noisier, including grammatical errors, ASR errors, dis-fluency, and more. The absence of reliable casing information also makes this a much harder task than written text. Third, the diversity of entities is higher in open domain chit-chat than in written documents, which usually focus on a few narrow topics. Fourth, users may recall incomplete or erroneous names of entities.

In addition to the spans of text with which could be entities, the detection model also predicts the categories of the spans. The categories limit the scope of the linking targets, e.g., only movie names or only character names, making linking more precise. We used an ensemble of models (experts) where each model is responsible for entity linking for a particular topic module. We used the LSH-Ensemble (43) model to ensure that linking time is short. The text span is linked to an entity based on the similarity between the text and the name of the entity.

We also designed a similarity metric function to ensure that linking is more robust against errors in the text. For instance, Google Knowledge Graph returns the song "Pink Fluffy Unicorns Dancing On Rainbows" for the query "unicorns" and returns the singer "Charley Pride" for the query "charlie pride" with different spellings. The latter is a valid matching while the former is not. To consider the similarity in the semantic level, we use BERT (7) to encode both the query and the returned result and compare their similarity in the hidden states. In addition, we consider the context for the hidden representation. Specifically, we feed the system and user utterances as consecutive sentences to the BERT model and get the mean hidden representation of the entity tokens to represent the query. Similarly, we concatenate the returned phrase (e.g. "Red Dead Redemption 2") with its description (e.g. "Survival game") and get the mean hidden state of all the tokens to represent the knowledge embedding. We use cosine similarity, F1, and F2 distances as the similarity metric. Cosine similarity with a pre-defined threshold works the best in our empirical analysis.

3.3.7 Sentiment

Sentiment analysis is mainly used for three purposes: to detect user intent together with dialog act, to detect user engagement, and help user utterance acknowledgment explained in Section 6. Instead of the standard binary prediction, such as the IMDb dataset (18) and SST-2 (28) or fine-grained five-class prediction, such as SST-5 (28), three classes of positive, negative, and neutral are more critical in our system understanding. Therefore, we map SST-5 into three classes in the sentence level and train a BERT (7) model. The model reaches an accuracy of 84.38%.

3.4 Dialog Management

We created a two-level hierarchical dialog manager to handle user inputs. The high-level system selects the best dialog module for each user request leveraging the output from NLU. After that, the low-level system activates the selected dialog module to generate a response.

3.4.1 High-Level System Dialog Management

High-level dialog manager *module selector* selects a dialog module to handle user utterance by detecting user's *intent* and referring to the *dialog module state* of the previously selected module.

Intent Classifier We define *functional intents*, *strong topic intents* and *moderate topic intents* to classify user utterances based on the human-bot conversations we collected during the competition. We break the user utterances into smaller segments, and each segment may have multiple intents.

- **Functional Intents**:
  - **Incomplete and Hesitant Utterances**: Due to endpointing model accuracy constraints in the Alexa ASR system, sometimes user utterances may be incompletely captured when a user has paused. To address this issue, we ask users to repeat when their utterances are detected as incomplete (eg. "I think it's") or hesitant (eg. "let me think").
  - **Clarification Intent**: If a user asks the bot to repeat (eg. "what did you say"), the system repeats its last response.
  - **Device Task Request**: Our system detects Alexa device system requests (eg."volume up") by guiding users to exit social mode to enable device functions. If the request is similar to a topic our system can handle, (eg. "play songs by Taylor Swift"), the system will ask the user if they are interested in talking about Taylor Swift.
- **Strong Topic Intents**:
  - **Topic Switch Intent**: If the user explicitly expresses their disinterest in the current topic (e.g. "I don't want to talk about movies anymore") or wants to change a topic (e.g. "can we talk about something else"), the system proposes another topic or asks what topic user wants to talk about.
  - **Topic Request Intent**: If the user specifies a topic with a command, e.g. "let's talk about movies", the dialog module selector immediately selects the movie module.
- **Moderate Topic Intents**:
  - **Topic Preference Intent**: Our system detects users' preference for a proposed topic to decide whether we should continue talking about it. For example, if system asks "Do you like to watch movies?", and user replies "I'm not a big fan of movies", the system will not continue the proposed topic even if user doesn't explicitly reject it by saying "no".
  - **Topic Intent**: Our system has a hierarchical method to detect topic intents in user utterances. First, based on all the dialogs we collected, we created a database to map user utterance to our topics. Since this database is human-annotated, it has a lower chance of false-positive topic detection. Thus, we use it as the first level to detect topics. In case we do not detect topics in the first level, we combine Google Knowledge Graph, Microsoft Concept Graph, and Cobot's built-in joint Dialog Act/Topic Model as our second level to detect topics. We tune the confidence thresholds of these three detectors based on the heuristic and dialog data we collected. One disadvantage of this second level is that it detects many false positives in the utterance as movie names, book titles, and song names. To avoid such false positives, we do not map detected movie names, book titles, and song names to topic modules.

Dialog Module State The previously selected module would be in one of the three state: "`CONTINUE`", "`UNCLEAR`" and "`STOP`". If the module has not finished its conversation, it will set itself to "`CONTINUE`" state, and the module selector will select it again if no functional intents or strong topic intents are detected. If a module detects that the user is talking about other topics, the topic module will attempt to propose another module that can handle the user's interests. It confirms whether the user wants to switch topics and sets itself as "`UNCLEAR`" state. If the user agrees, then the system switches the topic. If no clear intent or no other topics are detected, the system will select the same module again. When the conversation reaches a point where the current module has exhausted most of its predefined topics, the module will set "`STOP`" state to stop the current topic, and then the system proposes the next topic in the same turn.

Dialog Module Selector To select the most appropriate module, we set priorities for the detected intents. For functional intents, we select a functional module. For strong topic intents, we switch to the corresponding topic module no matter which states the previous module is in. Otherwise, we either select the previous module or propose a new topic depending on the state of the previous module. When the previous module is in "`UNCLEAR`" or "`STOP`" state and multiple topic intents are detected, if one of them is the same as the previously selected topic, we prefer to select this topic over others. Otherwise, we would propose the topic with the highest priority based on the personal topic order described in 5. In the next turn, if the user accepts the topic, our selector would select the proposed module.

Error Handling To handle system or dialog module crashes, we designed an error handling method based on the dialog act and fine-grain intents to keep the user engaged. For example, if the dialog module crashes when we detect fine-grain intents as "ans_like", we would say "Awesome, I like that too. I guess we have similar tastes" to continue the conversation. If the crash happens at the system level without any available NLU feature, the bot would say "My bad, I lost my train of thought. Do you want to try again, or would you rather talk about something else? " This way, users know something went wrong on our side, and it also gives users a chance to change the topic. In addition, in our user studies, we noticed that when the users give long utterances while the internet is not very stable, the Alexa Skill Kit server fails to get user input events and thus triggers a reprompt. Our bot sets reprompt template as "Thanks for sharing, Your thoughts are really interesting, but I'm struggling to keep up. Can you explain that again more simply for me? " to guide the user to speak a shorter utterance to avoid the same issue happening again.

3.4.2 Low Level Dialog Management

The Low-level dialog management consists of several dialog modules. There are two types of dialog modules. *Functional dialog module* handles functional intents such as device task requests. Each *Topic dialog module* handles one major topic, such as movies, music, and sports. Each topic dialog module has its own dialog flow design and uses *Finite State Machine* to manage dialog flow.

Topic Dialog Modules We keep the most popular topic modules from Gunrock, including movies, books, music, games, animals, sports, food, travel, news, technology and science, and retrieval. We improved their dialog design and extend the content as described in Section 4. We also add new modules users are interested in, for example:

- **Daily life and Outdoor Activities**: These two modules handle high-frequency topics users mention when answering open questions like "What do you like to do in your free time? ". Daily life covers topics in users' everyday lives, including friendship, family, relationships, playing with mobile phones, and chores. Outdoor activities covers hobbies such as biking, hiking, and kayaking. We start with a positive acknowledgment to increase the engagement (e.g., "It's really nice that you spend time playing with your kids. You must be an awesome parent!"). At the end of the topic discussion, we propose a relevant topic module to make the transition more natural and coherent. (e.g., "do you have a family pet? ").
- **Fashion**: The Fashion module discusses niche topics, including makeup, shopping for clothes, and fragrances. User adaptation to gender and agreeableness is implemented to propose more engaging sub-topics and respond to the user's interest level. The Fashion module was developed primarily for feminine audiences, while still containing neutral topics to adapt to a broader audience. In addition, a recommendation feature was developed that elicits user preferences to match them with an appropriate fragrance based upon their likes/dislikes. More detail is described in Section 5.4.
- **Comfort**: The comfort module is selected when users reveal they have had a terrible day or are having a hard time. We design special acknowledgment and validation for various conditions such as loneliness, worries, and sickness. After listening to the user's feelings and experience for several turns, it then proposes funny content like telling jokes to cheer the user up.

Finite State Machine (FSM) Our topic modules use a finite state machine (FSM) to represent each dialog state and handle user responses. We choose to use a finite state machine because it allows

greater control of the dialog flow so that the focus of engineering and experimentation can be on the dialog design and upstream NLU pipelines.

We create a custom FSM manager that provides the machine for state transition and a lightweight class structure for designing the modules' dialog flow. The FSM manager is state-centric, meaning transitions are defined within states. This design is chosen since traditional top-down approaches in designing FSMs are inflexible and hard to modify for our usage. Dialogs are dynamic and varied, and we emphasize acknowledging users' responses, which would be hard to encode using traditional methods such as transition tables.

Each state is identified by a unique string, which is used to define each dialog transition. A state also encapsulates all or part of the logic that should be executed in each dialog turn. Thus, each state can be loosely defined to represent a complex dialog turn, or part of the utterance depending on the clarity and reusability for each dialog flow. It can make API requests, generate template responses, or decide the next state transition based on the context. Each state can signal the FSM manager to perform a transition by returning the target state's identifier, and whether such a transition should happen in the current turn or the next. If the transition happens within the current turn, the system can chain states to create more complex responses. These transitions are defined inside the state because other logic, such as template generation and API queries, are defined within the state. Grouping them makes each state the only source of what should happen during a dialog turn. We find this makes designing our dialog flow more efficient and allows the reuse of parts of an utterance for similar responses. We also find defining the next state transition alongside the logic is more scalable than separating them since we only need to refer to each state to inspect the system's decision at each turn.

For each state, a tracker object is used to wrap around stateful storage and upstream NLU components, which can be passed between states and assign variables within the dialog turn or throughout the conversation. This tracker object allows us to encapsulate the upstream NLU pipelines and contextual information for user adaptation. Since the dialog flow is highly dependent on the upstream NLU results, this layer provides flexibility in our implementation to change and scale either efficiently.

Question Handling All the topic dialog modules use a common question handler to handle general questions. The question handler uses our custom back-story database to answer questions associated with our bot's persona, such as *What is your favorite color?*, and uses Amazon's EVI service[11] to handle factual questions. If there are no answers from both of them, we use our acknowledgment generator to generate a specific rephrased answer to express that the bot does not have an answer as described in 6. If the acknowledgment generator fails to generate a response, we then generate a different response based on the dialog act. For example, if the dialog act is "`OPEN QUESTION OPINION`", we then reply with "Good question, I haven't thought about that before. " to maintain an engaging conversation even we do not have an answer.

3.5 Natural Language Generation (NLG)

Our NLG module is a mixture of template-based and neural generation methods. It selects a manually designed template and fills out specific slots with information retrieved from the knowledge base by the dialog manager. It uses neural generation for unfamiliar topics and follow-up requests. We also use Amazon Speech Synthesis Markup Language (SSML) to provide our generated responses with prosodic variations.

3.5.1 Template Manager

The template manager centralizes all response templates for our system. Its design is similar to Gunrock. By consolidating all templates, we can avoid having duplicate responses. Each response is identified by a unique key, under which are templates of varying surface forms to make our content more natural and human-like. We use a shuffle bag method to ensure no templates are repeated after another.

Each template may also contain slots for dynamic substitution. This setup allows the insertion of retrieved data, such as weather or metadata of a movie. It also allows the insertion of captured NER from the user's response when generating acknowledgment.

[11] `https://www.evi.com/`

We have expanded the capability of the template manager by allowing key-values to be encoded alongside each template. This allows for more information to be retrieved per template usage. For instance, a musician specific template can be stored with key-value pairs describing the musician, its identifier in our knowledge base, and a name that is formatted for the speech synthesis. This grouping allows for better efficiency when using the template response.

3.5.2 Neural Generation Model

We believe that a dialog system, unlike humans, should be able to handle any topic when it has access to a large database or internet instead of responding generic utterances or changing to familiar topics. Therefore, we designed a retrieval module to retrieve relevant information from online sources similar to Gunrock (4; 36). However, the retrieval method is not robust with handling follow-up questions and comments. Therefore, we need a neural generation model to have a conversation about the retrieved information.

Text generation models are known for problems, including generating repetitive and non-specific sentences (16). Recent large pre-trained language models such as GPT-2 (22) and models fine-tuned for dialog domain such as DialoGPT (41) and Meena (2) achieve good performances on open-domain dialog. However, they still face the challenges of storing all knowledge into the model weights, thus cannot generate reasonable responses for unfamiliar topics, and the generated text highly depends on the training data. In addition, the generation process is not controllable. Another line of research is to collect dialog dataset based on pre-defined knowledge such as Persona-Chat (39), Wizard-of-Wikipedia (WoW) (9), and Topical-Chat (11). Models trained on these datasets are potentially more controllable by conditioning on specific attributes, but due to the nature of data collection, the collected utterances strictly follow the given knowledge and become less natural in conversation. More importantly, they rely heavily on the quality and relevance of the given attributes. To utilize the best of both worlds, we propose to inject knowledge into the pre-trained large language model. Specifically, we use DialoGPT-medium as our base model for latency concerns. We finetune the base model on the WoW dataset with KL-divergence. Similar to our retrieval method, we use Reddit as our database and use the retrieved knowledge as the condition if there is relevant information. The model learns to copy the knowledge and paraphrase the information. For instance, given the knowledge, "Covid-19 has infected more than 1.85 million people and killed at least 114,000 worldwide, according to Johns Hopkins University." and the context "... USR: I'm pretty good. Just a little bit worried about the coronavirus recently" in part of the conversation, the model generates "Covid-19 is a virus that has infected more than 1.85 million people, it scares me a lot" which rephrases the information correctly.

Even though DialoGPT achieves good performance on open-domain dialog, the responses are not conversational and may generate ungrammatical sentences due to its training data. Therefore, we also deploy the Blender (25) encoder-decoder model with 2.7B parameters trained with collected human conversational data (39; 23; 9) for more engaging response generation. Despite that it may still hallucinate knowledge (25), we mainly use this generation model for two purposes, question handling (such as responding to the user utterance "what about you what do you think about sukiyaki") and unfamiliar topic handling including follow up comments (such as responding "I love candy too! I love all kinds of candy, especially chocolate" to the user utterance "I like talking about candy I like candy"). In specific, because of the large number of layers in the model and the auto-regressive nature of generation, we use a beam size of one and a minimum beam length of five and ten, respectively, for question handling and unfamiliar topic handling to reduce the latency to a reasonable scope (less than one second end to end). In addition, since the training data is human-human conversations, we apply a rule-based filter to clean utterances that demonstrate a human personality such as "I like to eat at home with my wife", which clearly contradicts the personality of the Alexa social bot. Moreover, if the generated response is truncated due to latency issues or has an undesirable dialog act, we use the original question handler and retrieval responses instead.

3.5.3 Prosody Synthesis

Based on the analysis of adjusting speech synthesis to make our response more human-like (36), we utilize Amazon SSML[12] to format our responses before TTS. Specifically, we follow Gunrock to add fillers, insert pauses, and change speech speed based on the context of the utterance.

4 Dialog Flow

4.1 Improved Dialog Flow

In order to create an engaging conversational experience for the user, we have improved our dialog flow design both on the system level and topic module level.

4.1.1 System Level Dialog Flow

Topic Proposal and Transition When the current topic conversation comes to an end, the next topic can be proposed either by the current dialog module or the system. To make a seamless transition between modules, we set "propose_topic" (corresponding to dialog modules) and "propose_keywords" (keywords users mention) as global attributes to enable module-system and module-module communication.

If the current dialog module detects keywords that might fit another topic module better (eg. when user mentions "corona virus"), it sets state to "`UNCLEAR`", "propose_topic" to "`NEWS`","propose_keywords" to "corona virus", and confirms with the user if (s)he wants to talk more about corona virus. If the user agrees, the News module then seamlessly follows up on the specified keywords at the next turn.

The current module can also propose the closest topic after it finishes its sub-topic dialog flow. For example, after discussing users' opinions about the football team "Seattle Seahawks", Sport module can send the topic "Seattle Seahawks" to News to continue on the topic.

If current dialog module set state to "`STOP`" without setting "propose_topic", the system then proposes the next topic as described in Section 5.

4.1.2 Topic Module Dialog Flow

Movie From our user studies, we realize most users enjoy the movie trivia we provide, but some do not like how we repeatedly ask them to think of a movie to talk about. To address this, we propose a movie if the user cannot give an input or whenever we finish talking about a movie. The proposed movie is either a popular movie or a movie similar to the previously discussed one if they have common IMDB keywords (e.g., "space", "family", etc.) By interleaving between asking and proposing movies, we reduce user's cognitive load in the conversation and provide useful movie recommendations that users might be interested to know.

Music Music module in last year's Gunrock was mainly focused on pop music and singers. From the open question that we ask in system-level (5.2), we find that many users enjoy playing instruments, especially piano and guitar. We design flows discussing users' instrument learning experiences and recommends popular instrument artists to the users.

Games We have included a subtopic for discussing certain popular games in depth. For instance, if the user wants to talk about a particular game with a subtopic, we can discuss these games in-depth with carefully designed templates and questions. For example, if the user wants to talk about Animal Crossing, we can discuss specific information, such as costs of house upgrades, that would interest the user. We also adjust the module to recommend selected games at the beginning or when the user has no preference in any of the subtopics. This allows us better control of the dialog flow while providing in-depth content that crafted templates can offer. Finally, if the user mentions a game that we do not have information or templates for, we ask the user to provide a free-form description of the game. We then use different acknowledgments to show the system understands the user's response. This allows us to handle unknown entities while giving users a sense of control of the dialog.

[12] `https://developer.amazon.com/en-US/docs/alexa/custom-skills/speech-synthesis-markup-language-ssml-reference.html`

Food Built on top of food modules from Gunrock, we redesign the dialog flow to further adapt to user's interest in different subtopics in food. Our bot can talk about cooking, cuisines, and specific dishes or food that users like with carefully designed topic questions for each subtopic. When users do not provide specific interest to any subtopic, we introduce subtopics in the order of general interest to the audience in the food domain. For example, we initialize the conversation with questions like "what's your favorite dish?" that are more relevant to the majority of users and then gradually propose topics like "growing a garden" and "healthy eating" that are likely of interest to a smaller group of people.

News We revamped a large part of the news module this year to include more news content, and improved dialog flows to handle this content. We summarize the news articles and present them in multiple turns to better detect user intent and interest. We also include separate dialog flows for trending topics like coronavirus before presenting news about it to help make the flow more natural, engaging, and fluid. Also, we include more dialog flows containing debatable topics, which leads to a more engaging dialog. We experiment with a rule-based question generation model to ask the user if they want to continue with the news by generating questions for the next sentence in the news article. This ensures that the user is more informed about what will come next in the article before deciding if they wish to continue.

4.2 Extended content

Providing interesting content is always a priority for us to improve user experiences with our bot. This year, we continue to spend a great effort to collect high-quality web information to discuss with our users in different sub-modules.

People Also Ask People Also Ask [13] is a function from Google which provides a series of questions related to the search term that users have queried. Along with the list of these questions are the summarized answers that Google has extracted from different web pages. This function allows us to provide some insights to the topic word that people are interested in. We scrape the People Also Ask questions along with their answers for several popular topic words in the Food module as the first step to evaluate this data resource. If a user indicates interest in these topic words when talking to our food module, for example, "I like eating steaks," our bot will use the content from People Also Ask to provide interesting facts about the topic word. Instead of directly presenting a fact about the topic word, we create a more interactive way by asking if the user is interested to know the answer for a certain question from People Also Ask, such as "what is the best steak restaurant in the united states? " and "Is steak good for health? ". If users are interested, we present the answer associated with the question to the user. We use this function as a point of reference when designing the dialog flow for our sub-modules.

I Side With I Side With [14] is a website that allows people to vote on various debatable topics and share their opinions about them. We use these topics, content, and polls to create engaging dialogue flows in the News module sharing opinions from both sides of the debate while acknowledging and respecting user's personal opinions. We collect top pro and con opinions for the topic to present to the user.

5 Adaptation

5.1 User Profile

Our system keeps track of the user's profile and stores static data like user name, predicted gender, and dynamic data, including preferred topics and conversation style. The user name is stored when we greet the user by asking their name. Given the name, we predict user's gender based on a popular name database from U.S. Social Security Administration (SSA)[15], which provides baby names from 1980 to 2018 with gender information. The preferred topic is collected when users reveal their

[13] https://www.internetmarketingninjas.com/blog/search-engine-optimization/googles-people-also-ask-related-questions/

[14] https://www.isidewith.com/polls

[15] https://www.ssa.gov/oact/babynames

interests when answering the open questions described in 5.2. To learn users' conversation style, we dynamically update and store the user's dominant turn ratio described below.

5.2 Open Questions

An important goal of our system is to increase user engagement by finding the topics that users are interested in. Previous systems start a conversation by proposing one or several topics directly. It is rigid, unnatural and may not adapt well to different users' interests. In our system, we propose natural open questions when we start a new conversation to attract the user's attention as well as to know more about the user's interests. Based on the user's response, we then select the closest topic if possible.

Open Question Categories: We design three types of open questions: "`ASK_HOBBY`", "`PAST_EVENT`" and "`FUTURE_EVENT`". "`ASK_HOBBY`" is a batch of open questions that ask users' hobbies (e.g., "What do you like to do for fun?"). "`PAST_EVENT`" is a batch of open questions that ask users' past experiences (e.g., "What was the last thing that made you smile?"). "`FUTURE_EVENT`" is a batch of open questions that ask users' future plans (e.g., "What are your plans for the weekend?"). We leverage these natural open questions to start the conversation to get users' preferred topics.

Open Question Handling: When the user answers open questions, we use the module selecting strategy in Section 3.4.1 to select a topic module to handle it. If the system does not have an appropriate topic module, we will acknowledge the user's answer, as explained in Section 6, then naturally propose a topic for the user. On the other hand, if the user mentions multiple topics that our system can talk about, the system will save these topics as `preferred_topics` in the user profile. When the system is about to propose the next topic, these topics are given higher priority since the user already revealed interests in them.

Open Question vs. Topic Proposal: Since sometimes users do not mention all their interests in the first open question, our system will ask different types of open question after the user finishes talking about a topic. This way, we get more chances to know about the user's interest and select the corresponding topic accordingly. If the system already collected more than one `preferred_topics`, it will not ask the next open question until all the preferred topics are discussed.

Submissive Users vs. Dominant Users: We observed different types of user conversation styles in our system. Submissive users tend to follow the dialog flow initiated by the system, whereas dominant users like to direct the conversation and are more willing to express their opinions. Based on this observation, we have different strategies to fit their conversation style. Specifically, we define dominant turn ratio, calculating how many turns user utterance contains "`QUESTION`", "`COMMAND`", "`OPINION`"and "`STATEMENT`" dialog acts over all turns. If the ratio is higher than a threshold, the user is classified as a dominant user. We ask them open questions during topic transition since they are more willing to express their thoughts. On the other hand, we directly propose topics for submissive users as the open question may give them a high cognitive load and decrease their engagement. The ratio is updated every turn, and thus, it can adapt to the user's current behavior.

5.3 Next Module Selection

When the system finishes a topic and transits to the next module, our module selector takes three topic lists into consideration:

- `preferred_topics`: A list of all the topics that the user mentions when answering the system's open questions. These topics are likely to attract users' attention.
- `personal_topics`: A topic proposal order based on the user's predicted gender. Based on our collected conversations and heuristic experience, we curated different topic orders for males, females, and unpredictable genders.
- `used_topics`: A list of topics that the user has discussed or the system has proposed but was rejected by the user.

Our system would propose a topic that is in ''`preferred_topics`'' if it is not in ''`used_topics`''. If all the preferred topics have been discussed, we select the highest priority topic in ''`personal_topics`'' that is not in ''`used_topics`''. When all the topics have been used, we reset ''`used_topics`'' and propose the first topic in ''`personal_topics`''.

5.4 Adaptation in Topic Modules

Our system also adapts uniquely to each user in the topic module level.

In our Greetings Module, we first identify users by checking their user ids provided by Alexa Skill Kit. Then we use different templates to greet new users and returning users. For instance, if the user is identified as a new user, we would ask for their names; if they are a returning user, we would confirm if they are the same user by prompting for their names in case multiple users share the same device with their household. We also ask different open questions for returning users to know their recent updates (e.g., "What have you been up to recently? ") This adaptation humanizes our system by demonstrating our ability to recognize a user that has talked to us before.

In our Fashion Module, different subtopics are proposed based on inferred user gender. For example, we will propose makeup if we predict our user is female, and propose a different subtopic such as cologne if we instead predict our user is male. In addition, we choose different words based on user-gender. For example, for the female population, we refer to fragrances as *perfume*, while to the male population, we refer to fragrances as *cologne*. Furthermore, we categorize similar subtopics in the same group and track how often the user agrees or disagrees within the topic. If the user repetitively dismisses our response by saying "no", we promptly decide to transition to a different subtopic group that the user may prefer instead of staying within the same group.

6 Acknowledging User Utterances

Effectively signaling understanding is a key aspect of human conversation. According to Traum & Hinkelman (1992) (30), human interlocutors do not assume that they have been understood until their conversational partner explicitly acknowledges understanding, through either verbal or physical actions. Since our Alexa bot is unable to nod to indicate understanding, the bot must verbally acknowledge user utterances to fulfill its obligations as a conversational agent. For example, if our bot asks a user "What do you like to do in your free time?" and the user responds "I like to dance", it would be inappropriate for the bot to proceed with the conversation without in some way acknowledging the user's statement with a response such as "Ok, you like to dance." While a simpler response such as "Okay!" or "Nice!" may be appropriate in human conversation, we feel that a more explicit response which rephrases the user input implies fuller understanding by our conversational bot. Additionally, because detecting implied sentiment in user statements can be quite challenging, our rephrase strategy allows us to avoid positive acknowledgment of a potentially negative statement.

Similarly, we integrate a method to respond to user questions to which our bot is unable to generate a specific response. For example, if a user asks "What is my mother's name?", we are able to answer "I don't know what your mother's name is." Although we could simply respond "I don't know" to such a question, rephrasing and restating the user's query demonstrates to the user that, even if the bot is unable to answer the question, it at least understood what the user asked. According to Traum & Hinkelman (30), such a response, while not answering the question directly, grounds the question and communicates that the bot has understood. Like our statement acknowledgment, this question acknowledgment method makes our bot a more engaging conversational agent.

We implement a separate system module to generate utterances to acknowledge user statements and questions. We use the method described in and code provided by Dieter et al. (2019) (8) as a starting point for our acknowledgment system. Specifically, we implement a rule-based system as described in their paper, making modifications necessary to create variety and prevent unnatural repetition. We segment each of our sentences using our previously described sentence segmentation model. We then use a series of syntactic patterns and ordering rules to determine which segmented portion should be acknowledged. When necessary, we complete user utterances to resolve ellipsis using a rule-based method that considers the previous context and the syntax of the user response. For example, had the bot asked "What do you like to do in your free time", and the user responded "Dance", the sentence would be completed as "I like to dance". Input is then parsed using the Stanford CoreNLP (19) constituency parser, and utterances are reworded using a set of reordering rules. When responding to a question it is unable to specifically answer, the system prepends variants of "I do not know". Additionally, pronouns are replaced, and verb conjugations appropriately modified to generate a sensible and grammatical response.

In order to increase the diversity and relevance of the bot's responses, we also integrate Blender, as discussed in Section 3.5.2, to respond to certain user utterances. Blender is capable of generating meaningful responses to many user queries to which the bot does not have a specific response. For example, if the user asks "Have you ever heard about Roblox?", the rule-based method described in the preceding paragraph would respond "I don't know if I've ever heard about Roblox." Blender, on the other hand, will respond "I have heard of it, but I've never played it", which is clearly a more engaging response. At the same time, Blender output is often generic and not topical. For example, when a user asked the system "Who is Maddie?", Blender responded "I love movies", conditioned on the bot's previous utterance. The rule-based system, on the other hand, would respond "I don't know who Maddie is." We implement a rule-based system that determines if the Blender response contains any of the non-stop word tokens from the user input, to ensure that the Blender response is topical. If it does, we choose the Blender output over the generic rule-based acknowledgment.

In case the user intent is very common and occurs highly frequently, we use a hand-written acknowledgment template to acknowledge the user to improve user engagement instead of using the previous method to rephrase it. For example, if the user says "I don't know", the module generates "That's okay". If the user says "That's interesting!", it then replies, "I'm glad you like it! "

7 Results and Analysis

Figure 2 shows our system's rating over time for the past three months before Final Phase ended (June 4^{th}). Our chatbot's last seven-day rating improved from 3.60 to 3.75 during the last two months. We reached a median conversation duration of $2:14$ (min:sec) at the end of Final Phase, and lengthened our 90-percentile conversation duration by 24% (from $10:49$ to $13:31$) during the last three months. The increase in both ratings and duration reflects all the improvements we made in our system.

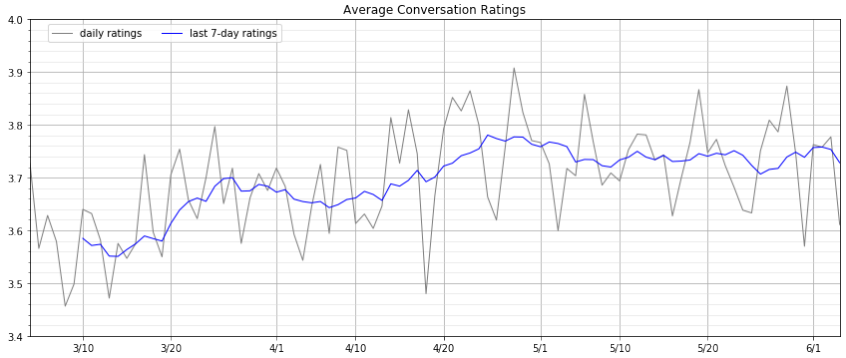

Figure 2: Average user ratings (daily and last 7-day) over time

7.1 Topic module performance analysis

We use the average rating per turn to evaluate the performance of each domain-specific module, where we assume each turn contributes equally to the rating of a complete dialogue. We chose this criteria over the average rating per dialog since each module's impact on the rating should correlate with the number of turns it hits.

Table 1 demonstrates the number of turns and average ratings for each topic module. *Movie* covers the most turns and has the longest average turn count per conversation, while *Fashion* and *Game* have the highest ratings. Figure 4 shows that the last 7-day rating of most modules improved over time in general. Among them, *News* increased the most as we adapt the dialog flow better to corona virus-related topic.

Topic Modules	Total Turn Count	Average Turn Count Per Conversation	Average Rating
Movie	87353	14.51	4.09
Animal	65672	12.80	4.13
Launch Greeting	55829	2.92	3.80
Music	36907	10.01	4.06
Game	31456	8.57	4.16
Social	28300	2.66	3.84
Sport	26948	9.44	4.12
Food	25117	10.80	4.09
Book	18774	12.33	4.06
Tech and Science	9774	8.08	4.13
News	8262	8.12	4.03
Travel	7877	8.29	4.09
Fashion	6174	6.14	4.18
Daily Life	4209	1.59	4.00
Comfort	2527	4.84	3.75
Outdoor Activities	1044	2.18	4.02
Self-Disclosure	897	2.49	2.27
Jokes	560	3.46	4.19
ALL	459751	23.98	4.02

Table 1: Overall performance of each topic module during the last month of Final (May 4 - June 4, ordered by total turn count)

7.2 Dialog Strategy Effectiveness

7.2.1 Topic entry analysis

To understand how users enter each topic module and evaluate if asking open question and proposing a topic are effective in helping users find topics that interest them, we calculate entry count of entry method for all modules, as shown in Figure 3. Almost all modules have some entries from open questions, which means that the open questions might be effective for nailing down topics for users. It is notable that Game and Movie(including TV) have the highest entry count from open question, implying that users are actively interested in talking about these topics.

In table 2, we compare the performances of different entries for all topic modules. For most modules, topic proposal entries have a better rating than open question entries. It may be because user utterances are generally relatively simple in topic proposal entry (e.g., answering "yes" to the question "Do you like animals?"). At the same time, they usually are more complicated in open question entries (e.g., saying "i like baking" to the question "What do you like to do for fun?"), which requires the corresponding module to detect entities and map them to the corresponding state. If the module does not have a pre-defined flow that fits that entity, the response might not meet the user's expectation and thus lead to lower ratings.

7.2.2 Topic proposal analysis

To figure out if users are interested in the topics we proposed, and how users accept the proposed topics, we measure the user acceptance rate as displayed in Figure 3. Animal and Music have the highest acceptance rate (0.75 and 0.72 respectively). The result indicates what topics interest the users the most, and can serve as a reference when deciding which topic modules the bot should propose first to meet most users' interests. Note that the acceptance rate does not have an evident correlation to the topic module rating. For example, Fashion module has the lowest acceptance rate while having the highest rating as it has an engaging dialog flow. In other words, modules with the highest rating might not have the highest rating if the dialog does not meet the user's expectation. This reveals that dialog flow plays an important role in the bot's overall performance.

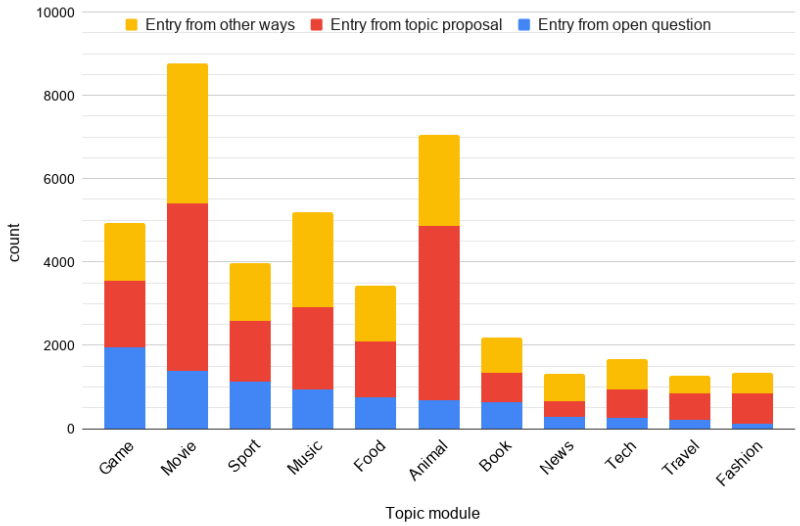

Figure 3: Distribution of different entry method for each topic module during the last month of Final (May 4 - June 4)

Topic module	Open question entry	Topic proposal entry	Other entry
Movie	4.05	**4.09**	4.08
Book	3.96	**4.20**	4.03
Music	4.07	**4.10**	4.01
Sport	4.01	**4.20**	4.18
Food	3.87	**4.24**	4.05
Animal	4.17	4.11	**4.18**
Travel	4.05	**4.19**	3.90
Game	**4.22**	**4.13**	4.13
Fashion	**4.26**	4.16	4.18
Tech and Science	4.11	**4.20**	4.16
News	3.88	**4.31**	3.90

Table 2: Rating of each topic module from different entries during the last month of Final (May 4 - June 4)

Topic module	User acceptance count	Topic proposal count	User acceptance rate
Animal	4192	5573	0.75
Music	1976	2739	0.72
Movie	4038	5818	0.69
Food	1332	2031	0.66
Tech and science	665	1071	0.62
Travel	629	1071	0.59
Book	686	1220	0.56
Game	1585	2930	0.54
News	371	795	0.47
Sports	1468	3220	0.46
Fashion	727	1804	0.40

Table 3: Average user acceptance rate for all topic modules during the last month of Final (May 4 - June 4)

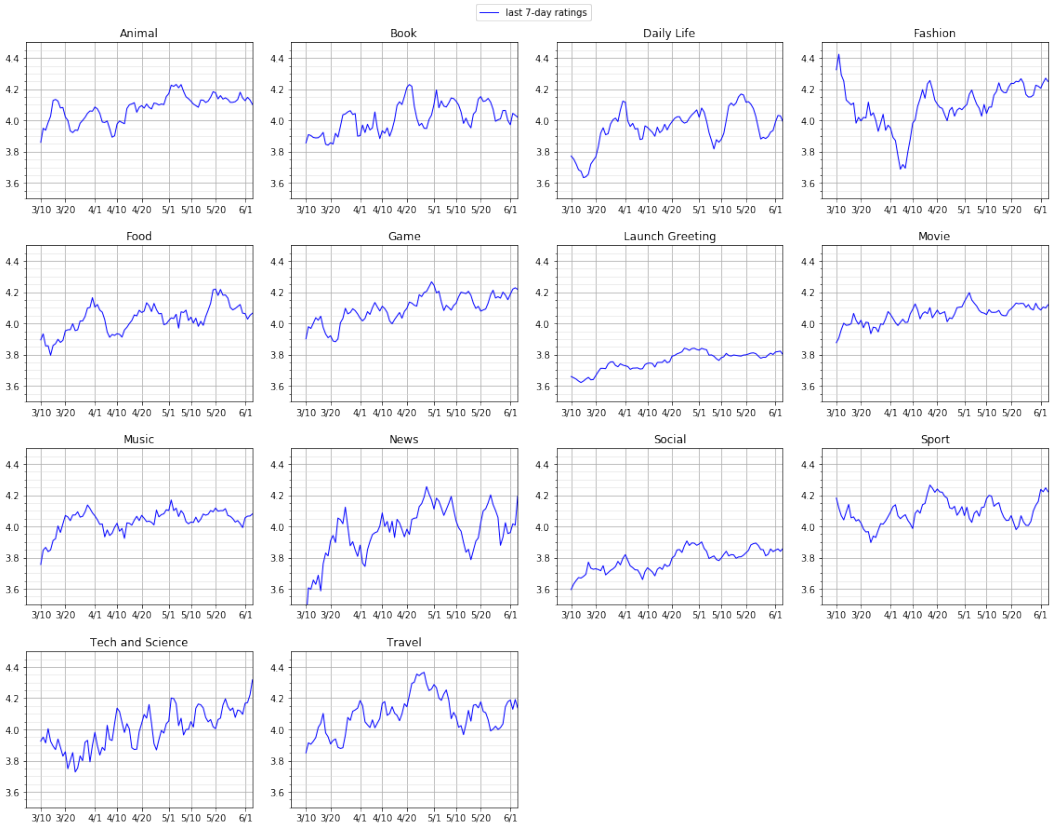

Figure 4: Average user ratings per module (last 7-day) over time

Utterance	**Noun phrases**	**Key phrases**
what year was uhhh julius caesar murdered	uhhh julius caesar	julius caesar
can we not talk about animals left in the wild	animals, the wild, wild	animals, wild
eat at a restaurant	a restaurant	at a restaurant
tom clancy's rainbow six siege	tom clancy's, six siege	tom clancy's rainbow six siege
tell her wings of fire	her wings, fire	wings of fire
my dream vacation would be toronto	dream vacation	dream vacation, toronto
oh yeah my mom and dad have taking me a bunch of places	mom, dad, a bunch, places	bunch, places

Table 4: Differences between noun phrases (extracted using the constituency parser) and key phrases (output of the KeyPhrase model). Examples are cherry-picked. Key phrases are usually more precise than noun phrases.

7.3 Name Entity Detection

We evaluated the difference between the noun phrases and the KeyPhrase outputs introduced in Section 3.3.5. Noun phrases are extracted from the output of the constituency parser with the utterance as the input. Key phrases are output from the KeyPhrase model.

We investigated the differences between noun phrases and key phrases (Table 4). Unlike written text, spoken utterances are noisy, so the noun phrases may include fillers. For example, in the utterance "what year was uhhh julius caesar murdered", the extracted noun phrase is "uhhh julius caesar" instead of just "julius caesar". In addition, extracting all noun phrases from a constituency parse may include overlapping phrases. For example, both "the wild" and "wild" are extracted from the utterance "can we not talk about animals left in the wild". Although including overlapping phrases may help with

reasoning in topic modules, it has a negative impact on latency since there is more work that needs to be done in the entity linking step. Additionally, key phrases capture more phrases with named entities such as “tom clancy’s rainbow six siege”, “wings of fire”, and “toronto”. However, key phrases sometimes miss certain phrases such as "mom" and "dad" in the last example in Table 4.

8 Conclusion

We designed an open-domain social conversation system that achieved an average rating of 3.73 out of 5 in the last seven days of the semi-finals. We made many contributions in entity detection and linking, user-adaptive dialog management, and integrating generation and template-based methods in a social chatbot. In sum, we increased its understanding by adding more powerful neural network-based models and improved dialog response by manually creating templates with free-form generation models. These generation models handle corner cases while carefully written templates handle popular intents. We also improved the selection of proposed topics for different users based on their profile information, such as gender and dialog act use distribution to create an adaptive and unique conversation experience for each user.

Acknowledgments

We would like to acknowledge the help from Amazon in terms of financial and technical support. We would also like to thank Leann Sotelo, Michelle Cohn for helping on revising the system dialog templates; Michel Eter for providing feedback to our system.

References

[1] ABEND, O., AND RAPPOPORT, A. Universal conceptual cognitive annotation (ucca). In *ACL (1)* (2013), The Association for Computer Linguistics, pp. 228–238.

[2] ADIWARDANA, D., LUONG, M.-T., SO, D. R., HALL, J., FIEDEL, N., THOPPILAN, R., YANG, Z., KULSHRESHTHA, A., NEMADE, G., LU, Y., AND LE, Q. V. Towards a human-like open-domain chatbot, 2020.

[3] BUDZIANOWSKI, P., WEN, T.-H., TSENG, B.-H., CASANUEVA, I., ULTES, S., RAMADAN, O., AND GAŠIĆ, M. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278* (2018).

[4] CHEN, C.-Y., YU, D., WEN, W., YANG, Y. M., ZHANG, J., ZHOU, M., JESSE, K., CHAU, A., BHOWMICK, A., IYER, S., SREENIVASULU, G., CHENG, R., BHANDARE, A., AND YU, Z. Gunrock: Building a human-like social bot by leveraging large scale real user data. In *2nd Proceedings of Alexa Prize* (2018).

[5] DANESCU-NICULESCU-MIZIL, C., AND LEE, L. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011* (2011).

[6] DAVIDSON, S., YU, D., AND YU, Z. Dependency parsing for spoken dialog systems, 2019.

[7] DEVLIN, J., CHAN, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805, 2018.* (2018).

[8] DIETER, J., WANG, T., CHAGANTY, A. T., ANGELI, G., AND CHANG, A. X. Mimic and rephrase: Reflective listening in open-ended dialogue. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 393–403.

[9] DINAN, E., ROLLER, S., SHUSTER, K., FAN, A., AULI, M., AND WESTON, J. Wizard of wikipedia: Knowledge-powered conversational agents. *CoRR abs/1811.01241* (2018).

[10] FANG, H., CHENG, H., SAP, M., CLARK, E., HOLTZMAN, A., CHOI, Y., SMITH, N. A., AND OSTENDORF, M. Sounding board: A user-centric and content-driven social chatbot. *CoRR abs/1804.10202* (2018).

[11] GOPALAKRISHNAN, K., CHEN, B. H. Q., GOTTARDI, A., KWATRA, S., VENKATESH, A., GABRIEL, R., AND HAKKANI-TUR, D. Topical-chat: Towards knowledge-grounded open-domain conversations.

[12] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural Computation 9*, 8 (1997), 1735–1780.

[13] KHATRI, C., HEDAYATNIA, B., GOEL, R., VENKATESH, A., GABRIEL, R., AND MANDAL, A. Detecting offensive content in open-domain conversations using two stage semi-supervision, 2018.

[14] KHATRI, C., HEDAYATNIA, B., VENKATESH, A., NUNN, J., PAN, Y., LIU, Q., SONG, H., GOTTARDI, A., KWATRA, S., PANCHOLI, S., CHENG, M., CHEN, Q., STUBEL, L., GOPALAKRISHNAN, K., BLAND, K., GABRIEL, R., MANDAL, A., HAKKANI-TUR, D., HWANG, G., MICHEL, N., KING, E., AND PRASAD, R. Advancing the state of the art in open domain dialog systems through the alexa prize, 2018.

[15] KONSTAS, I., IYER, S., YATSKAR, M., CHOI, Y., AND ZETTLEMOYER, L. Neural AMR: sequence-to-sequence models for parsing and generation. *CoRR abs/1704.08381* (2017).

[16] LI, J., GALLEY, M., BROCKETT, C., GAO, J., AND DOLAN, B. A diversity-promoting objective function for neural conversation models. *CoRR abs/1510.03055* (2015).

[17] LI, J., MONROE, W., RITTER, A., GALLEY, M., GAO, J., AND JURAFSKY, D. Deep reinforcement learning for dialogue generation. *CoRR abs/1606.01541* (2016).

[18] MAAS, A. L., DALY, R. E., PHAM, P. T., HUANG, D., NG, A. Y., AND POTTS, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Portland, Oregon, USA, June 2011), Association for Computational Linguistics, pp. 142–150.

[19] MANNING, C. D., SURDEANU, M., BAUER, J., FINKEL, J., BETHARD, S. J., AND MCCLOSKY, D. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations* (2014), pp. 55–60.

[20] PHILIPS, L. The double metaphone search algorithm. *C/C++ Users J. 18*, 6 (June 2000), 38–43.

[21] PICHL, J., MAREK, P., KONRÁD, J., MATULÍK, M., NGUYEN, H. L., AND SEDIVÝ, J. Alquist: The alexa prize socialbot. *CoRR abs/1804.06705* (2018).

[22] RADFORD, A., NARASIMHAN, K., SALIMANS, T., AND SUTSKEVER, I. Improving language understanding by generative pre-training.

[23] RASHKIN, H., SMITH, E. M., LI, M., AND BOUREAU, Y. I know the feeling: Learning to converse with empathy. *CoRR abs/1811.00207* (2018).

[24] REED, S., LEE, H., ANGUELOV, D., SZEGEDY, C., ERHAN, D., AND RABINOVICH, A. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596* (2014).

[25] ROLLER, S., DINAN, E., GOYAL, N., JU, D., WILLIAMSON, M., LIU, Y., XU, J., OTT, M., SHUSTER, K., SMITH, E. M., BOUREAU, Y.-L., AND WESTON, J. Recipes for building an open-domain chatbot, 2020.

[26] SHEN, Y., YUN, H., LIPTON, Z. C., KRONROD, Y., AND ANANDKUMAR, A. Deep active learning for named entity recognition. *CoRR abs/1707.05928* (2017).

[27] SHUM, H., HE, X., AND LI, D. From eliza to xiaoice: Challenges and opportunities with social chatbots. *CoRR abs/1801.01957* (2018).

[28] SOCHER, R., PERELYGIN, A., WU, J., CHUANG, J., MANNING, C. D., NG, A., AND POTTS, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (Seattle, Washington, USA, Oct. 2013), Association for Computational Linguistics, pp. 1631–1642.

[29] SUTSKEVER, I., VINYALS, O., AND LE, Q. V. Sequence to sequence learning with neural networks. *CoRR abs/1409.3215* (2014).

[30] TRAUM, D. R., AND HINKELMAN, E. A. Conversation acts in task-oriented spoken dialogue. *Computational intelligence 8*, 3 (1992), 575–599.

[31] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. *CoRR abs/1706.03762* (2017).

[32] VINYALS, O., AND LE, Q. V. A neural conversational model. *CoRR abs/1506.05869* (2015).

[33] WALLACE, R. The elements of AIML style. ALICE AI Foundation., 2004.

[34] WANG, X., SHI, W., KIM, R., OH, Y., YANG, S., ZHANG, J., AND YU, Z. Persuasion for good: Towards a personalized persuasive dialogue system for social good. *arXiv preprint arXiv:1906.06725* (2019).

[35] WU, Q., ZHANG, Y., LI, Y., AND YU, Z. Alternating roles dialog model with large-scale pre-trained language models, 2019.

[36] YU, D., COHN, M., YANG, Y. M., CHEN, C.-Y., WEN, W., ZHANG, J., ZHOU, M., JESSE, K., CHAU, A., BHOWMICK, A., IYER, S., SREENIVASULU, G., DAVIDSON, S., BHANDARE, A., AND YU, Z. Gunrock: A social bot for complex and engaging long conversations, 2019.

[37] YU, D., AND SAGAE, K. UC Davis at SemEval-2019 task 1: DAG semantic parsing with attention-based decoder. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (Minneapolis, Minnesota, USA, June 2019), Association for Computational Linguistics, pp. 119–124.

[38] YU, D., AND YU, Z. Midas: A dialog act annotation scheme for open domain human machine spoken conversations. *arXiv preprint arXiv:1908.10023* (2019).

[39] ZHANG, S., DINAN, E., URBANEK, J., SZLAM, A., KIELA, D., AND WESTON, J. Personalizing dialogue agents: I have a dog, do you have pets too? *CoRR abs/1801.07243* (2018).

[40] ZHANG, X., LI, C., YU, D., DAVIDSON, S., AND YU, Z. Filling conversation ellipsis for better social dialog understanding, 2019.

[41] ZHANG, Y., SUN, S., GALLEY, M., CHEN, Y.-C., BROCKETT, C., GAO, X., GAO, J., LIU, J., AND DOLAN, B. Dialogpt: Large-scale generative pre-training for conversational response generation, 2019.

[42] ZHAO, T., ZHAO, R., AND ESKÉNAZI, M. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *CoRR abs/1703.10960* (2017).

[43] ZHU, E., NARGESIAN, F., PU, K. Q., AND MILLER, R. J. Lsh ensemble: Internet-scale domain search. *Proceedings of the VLDB Endowment 9*, 12 (2016).